RECEIVED
04 OCT 2004
WIPO PCT

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference PWO-0872	**FOR FURTHER ACTION**	See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)
International application No. PCT/CA 03/00453	International filing date *(day/month/year)* 28.03.2003	Priority date *(day/month/year)* 29.03.2002
International Patent Classification (IPC) or both national classification and IPC H04L29/06		
Applicant RESEARCH IN MOTION LIMITED		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of 8 sheets, including this cover sheet.

   ☒ This report is also accompanied by ANNEXES, i.e. sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

   These annexes consist of a total of 1 sheets.

3. This report contains indications relating to the following items:

   - I ☒ Basis of the opinion
   - II ☐ Priority
   - III ☐ Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
   - IV ☐ Lack of unity of invention
   - V ☒ Reasoned statement under Rule 66.2(a)(ii) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
   - VI ☐ Certain documents cited
   - VII ☐ Certain defects in the international application
   - VIII ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
30.09.2003	01.10.2004
Name and mailing address of the international preliminary examining authority: European Patent Office - P.B. 5818 Patentlaan 2 NL-2280 HV Rijswijk - Pays Bas Tel. +31 70 340 - 2040 Tx: 31 651 epo nl Fax: +31 70 340 - 3016	Authorized Officer Wiltink, J Telephone No. +31 70 340-2969

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No. PCT/CA 03/00453

I. Basis of the report

1. With regard to the **elements** of the international application *(Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17))*:

Description, Pages

1-26	as originally filed

Claims, Numbers

1, 3, 5-10	as originally filed
2, 4	received on 20.09.2004 with letter of 20.09.2004

Drawings, Sheets

1/25-25/25	as originally filed

2. With regard to the **language**, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.

   These elements were available or furnished to this Authority in the following language: , which is:

   ☐ the language of a translation furnished for the purposes of the international search (under Rule 23.1(b)).

   ☐ the language of publication of the international application (under Rule 48.3(b)).

   ☐ the language of a translation furnished for the purposes of international preliminary examination (under Rule 55.2 and/or 55.3).

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

   ☐ contained in the international application in written form.

   ☐ filed together with the international application in computer readable form.

   ☐ furnished subsequently to this Authority in written form.

   ☐ furnished subsequently to this Authority in computer readable form.

   ☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

   ☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.

4. The amendments have resulted in the cancellation of:

   ☐ the description, pages:

   ☐ the claims, Nos.:

   ☐ the drawings, sheets:

5. ☐ This report has been established as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed (Rule 70.2(c)).

   (Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.)

6. Additional observations, if necessary:

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Yes:	Claims	1-10
	No:	Claims	
Inventive step (IS)	Yes:	Claims	
	No:	Claims	1-10
Industrial applicability (IA)	Yes:	Claims	1-10
	No:	Claims	

2. Citations and explanations

   see separate sheet

Re Item V

Reasoned statement under Rule 66.2(a)(ii) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1 Reference is made to the following documents:

D1: "Fast Dispatch for Stock Hardware", John R. Rose, OOPSLA Proceedings, September 25-30, 1988. Special Issue Of SIGPLAN Notices, Vol. 23, No. 11, November 1988

D2: US-A-5889995, "Using Constant Selectors For Method Identification", Segnan, M., 30 March 1999

2 The document D1 is regarded as being the closest prior art to the subject-matter of claim 1 and insofar as this claim can be understood, this document shows the following features thereof (the references in parentheses applying to D1):

- method for handling runtime objects operating on a wireless device (run-time type checking and/or method selection, page 27, right-hand column, lines 1-13; limited physical memory, page 27, right-hand column, lines 41-44)
- runtime objects are instantiations of classes, classes can implement methods (applies to fully object-oriented languages, page 27, right-hand column, lines 5-8)
- for each class C, for all their methods M, create a tuple (ord(M), &M) where ord(M) is the ordinal value of the method M and &M is a reference to the implementation of M in a class C (tag and key, page 29, left-hand column, lines 1-12)
- using (ord(M), &M) to dispatch a method to the implementation of the runtime object (dispatch functions are implemented, page 29, right-hand column, lines 15-36)

The method of document D1 thus achieves fast and direct method dispatching for methods implemented by a class in a class hierarchy, without a complex and elaborate search through the class hierarchy.

2.1 The subject-matter of claim 1 therefore differs from this known D1 in that:

- classes can implement interfaces having methods
- for each class C, for all their interfaces I, creating a tuple (ord(I), &I) where ord(I) is the ordinal value of the interface I and &I is a reference to the implementation of I in a class C

- using (ord(I), &I) to check if a runtime object is an instance of a class that implements I

The effect of these extra features is that methods can be directly invoked in an interface hierarchy, using a single intermediate look-up step, without a complex and elaborate search through the interface hierarchy.

The problem to be solved by independent method claim 1 may therefore be regarded as "how to support direct method-invocation in object-oriented languages supporting interfaces (such as Java)?"

The solution proposed in claim 1 of the present application cannot be considered as involving an inventive step (Article 33(3) PCT) for the following reasons:

Given the use of dispatch tables as disclosed in D1, building an index to all interfaces of a class as well as the methods implemented by a class, is obvious to a person skilled in the art of object-oriented programming language design and implementation. The use of the same table and tuples for tracking interfaces as those used in D1 for methods, is an obvious design choice and involves no inventive step in the sense of Article 33(3) PCT.

3 Dependent claims 2-8 do not contain any features which, in combination with the features of any claim to which they refer, meet the requirements of the PCT in respect of novelty and/or inventive step (Article 33(2) and (3) PCT), the reasons being as follows:

3.1 The technical features of claim 2 are:

- check if runtime object can be cast into type of an interface I by providing class data for runtime object to determine class type
- using (ord(I), &I) to check if runtime object is instance of I

Getting the type of a run-time object is a built-in method in all object-oriented languages, and is therefore not considered to involve an inventive step. Using a table of interfaces to check if interface I is on the list is trivial to a person skilled in the art. When faced with the problem of checking if something is on a list, building a table with an identifer (ord(I) in this case) as an index to this list, is well-known (D1, page 29, left-hand column, lines 1-22). Claim 2 is therefore not inventive (Article 33(3) PCT).

3.2 The technical features of claim 3 are:

- dispatch method call by providing class data for runtime object to determine class type
- using (ord(M), &M) to directly dispatch the method call

By analogy to claim 2, claim 3 is not inventive (Article 33(3) PCT).

3.3 The technical features of claim 4 are:

- compiler system generates check and dispatch information for an input set of classes and interfaces
- information includes interface ordinals ord(I), associated interface references (&I), method ordinals ord(M) and associated method references (&M)
- wireless device includes runtime storage to store (ord(I), &I)

Claim 4 is not inventive because, because D1 clearly shows that the creation of tuples of ordinals and references is known in the art (D1, tag, key, page 29, left-hand column, lines 1-22). A compiler process for generating method invocation instructions (check and dispatch information) for a source-code program expressed in an object-oriented computer programming language, that can be implemented as a compiler or a set of utilities and a compiler (D2, compiler, page 1, abstract). While D2 does not disclose also including information about interfaces but only describes information about methods, the inclusion of interfaces into the information generated by the compiler is not inventive.
It is also generally known in the art that wireless devices have runtime storage at their disposal, either on a SIM card or built-in in the phone's hardware. Using this storage to store the tuples (ord(I), &I) is a trivial design decision. Therefore, claim 4 is not inventive (Article 33(3) PCT).

3.4 Claim 5 is not inventive by analogy to claim 4. If storing the interface tables in the runtime storage of the wireless device is trivial, then storing method tables there as well obviously follows from that. Therefore, claim 5 is not inventive (Article 33(3) PCT).

3.5 Claim 6 describes the technical features:

- runtime storage
- runtime context including objects for which a direct check is performed
- runtime context includes method calls for which a direct dispatch is performed

Runtime storage is trivial in the art of computers and computer programming, and therefore not considered to involve an inventive step.
It is also apparent to a person skilled in the art of computer programming that objects on which actions are performed or have been performed, will be stored (temporarily or persistently) in some kind of runtime storage space. This can be working memory, the processor's stack or heap, cache memory or even disk space. It is therefore trivial to include the objects for which a direct check is performed, as well as the method calls for which a direct dispatch is performed, in this runtime context of the runtime storage. Therefore, claim 6 is not inventive (Article 33(3) PCT).

3.6 Claim 7 describes the technical features:

- a wireless device with a runtime processor operating a Java-based environment
- the processor uses (ord(I), &I) to check if a runtime object implements an interface
- the processor uses (ord(M), &M) to dispatch the method of I to the runtime object

By analogy to the reasoning given for claims 2 and 3, the runtime type check to see if an interface is implemented by an object, and the runtime dispatch of a method, have already been shown not to be inventive. A wireless device with a runtime processor is readily known in the art and therefore not new. In the context of object-oriented programming languages, Java is an obvious and trivial choice of the person skilled in the art, especially when targeted to wireless devices. The subject-matter of claim 7 is therefore not inventive (Article 33(3) PCT).

3.7 The additional subject-matter of claim 8, a method call triggering the direct-check-and-dispatch steps claimed, is obvious to a person skilled in the art of object-oriented programming. The method call is the basis for all interaction between objects, local or remote, in an object-oriented system. Its use is not novel and does not present an inventive step in the sense of Article 33(2) and (3) PCT.

3.8 To the extent that claim 9 can be understood, this claim appears to formulate the method of claim 1 in the form of a system with appropriate data structures, which immediately correspond to the different method steps; claim 9 further specifies that the assignment of ordinals to interfaces and methods specify the constraints:

- two interfaces have different ordinals if there exists a class that implements

both said interfaces

- two methods have different ordinals if there exists a class that implements both said methods

Given that the trivial assignment a unique ordinal to each interface and method, irrespective of the implementation relationships, results in an assignment satisfying both constraints as formulated in claim 9, the inclusion of this limitation does not involve an inventive activity. For essentially the same reasons as given for claim 1, the subject-method of claim 9 is therefore not considered to involve an inventive step (Article 33(3) PCT).

3.9 Dependent claim 10, in light of D1 and D2, is also not inventive by analogy to the reasoning for claim 4 and claim 9 (Article 33(3) PCT). D1 shows that making hash tables for a class's methods and the addresses of the implementation is known. From that, it follows that doing the same for a class's interfaces is trivial. D2 shows a compiler creating such tables for a class's methods, and therefore doing the same for interfaces follows from what is known in combination with D1. Claim 10 is therefore not inventive (Article 33(3) PCT).

2. The method of claim 1, wherein a direct interface check is performed to verify if the runtime object can be cast into the type of an interface by performing steps comprising:

providing class data for the runtime object in order to determine what class type provided the constructor for the runtime object;

using the interface ordinals' association with the interface references to check whether the runtime object is an instance of an interface;

wherein an interface allows multiple inheritance to occur without multiple inheritance of implementation.

3. The method of claim 1, wherein a dispatch is performed by performing steps comprising:

providing class data for the runtime object in order to determine what class type provided the constructor for the runtime object;

using the method ordinals association with the method references to directly dispatch the method.

4. The method of claim 1, wherein a compiler system generates check and dispatch information based upon an input set of classes and set of interfaces,

wherein the check and dispatch information includes the interface ordinals, the interface references associated with the interface ordinals, the method ordinals, and the method references associated with the method ordinals, wherein the wireless device includes runtime storage to store on the wireless device the assigned interface ordinals and their associated interface references.